A Recursive Block based Power Aware Imprecise Multiplier

Eric Zhang Xinfei Guo
ECE 6332/4332 – Fall 2012
University of Virginia
<hz9xa, xg2dt>@virginia.edu

ABSTRACT

This paper proposes a novel recursive structure block-based Wallace tree multiplier with the capability to tune its accuracy based on the precision and power requirements. The multiplier performs multiplication operations recursively from small blocks to large blocks, with a fine grain capability of turning off small blocks to trade off with power. By tuning the fidelity design "knob", we show that the 32-bit implementation of the proposed imprecise multiplier can achieve an average power savings up to 47% and 10% improvement in latency compared to a convectional Wallace tree multiplier. A case study on JPEG compression algorithm demonstrates the promising application of such multiplier design.

1. INTRODUCTION

As the performance for today's multipliers increases thanks to the new materials, advanced fabrication technologies and various design strategies, however, the demand for power has been increasing exponentially while power remains a very limited resource in many battery-powered devices. In addition, multipliers are usually the most power-consuming unit since they have a longer latency and larger bit-width[1]. And in many applications such as multimedia and DSP, 100% accuracy is most of the time not a strict requirement due to the inherent limitation of human-beings and the inevitable occurrence of errors in VLSI technology[2]. By introducing accuracy as an additional design parameter, we can achieve a new Pareto optimal surface for the multiplier design.

Various design techniques for imprecise multipliers ranging from physical to architectural levels were proposed. In [3], an energy-aware probabilistic multiplier is proposed based on the idea of probabilistic CMOS. In [4], a low-power reconfigurable signed pipeline array multiplier is presented, they utilize partially guarded computation and truncated multiplication techniques to reduce power consumption. In [1] and [5], a low-power high-performance multi-precision multiplier is proposed. Dynamic input range detection is employed to examine the multiplier's operands. Voltage scaling and frequency scaling are also implemented. In [6], the authors present a novel multiplexer based truncated array multiplier, which has leveraged and improved upon three existing truncation algorithms. In [7], a variable-precision multiplier for equalizer with adaptive modulation is presented. Dynamic voltage scaling and gating techniques are proposed. In [8], a left-to-right array multiplier is proposed. It combines signal flow optimization of adder array, left-to-right leapfrog signal flow and reduction array splitting.

Most of these existing low power imprecise multipliers focus on the bit-wise truncation and the array multiplier structure. Bit-wise truncation can be flexible for fine grained precision control, but it is inefficient with significant area overhead in order to assign each bit a control signal. From [9], the Wallace tree structures show a roughly 10% power advantage over array-based designs. In our multiplier design, we utilize both array and tree structure. The array structure is employed in the small block multiplication and the partial products accumulation is implemented in Wallace tree structure. The small bit width array multiplier gives superior performance and can be gated accordingly to save power.

The rest of paper is organized as follows: Section 2 describes the general algorithm of the multiplier design. Section 3 illustrates the detailed circuit and hardware implementation. Low power strategies and analysis are described in Section 4. In Section 5, simulation results with comparison to corresponding conventional Wallace tree multiplier. Section 6 gives conclusions and future work.

2. SYSTEM OVERVIEW

2.1 Architecture

The proposed block based multiplier is designed for n by n bits multiplications including 8 by 8, 16 by 16, 24 by 24 and 32 by 32. In this paper, we will present 8 by 8 and 32 by 32 bit multiplications since the structures can be extended in an easy way. For two n-bit inputs a<n-1:0> and b<n-1:0>, we divide both a and b into k blocks, and each block contains n/k bits. Then a parallel multiplication will be performed among each block. The

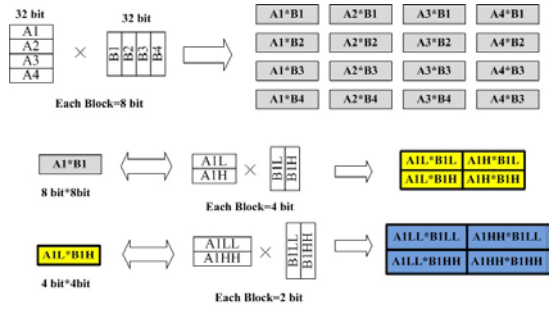


Figure 1 Recursive Structure block multiplication

resulting block products will be in a k-by-k matrix. Figure 1 gives an example of a 32 by 32 bit multiplier that is divided into 4 blocks (k=4), resulting in a 4 by 4 square matrix. And each element is a 8 by 8 bit multiplier. Then the 8 by 8 bit multiplier can be further divided into 4 by 4 and then 2 by 2. The resulting 2 by 2 bit multiplier will be implemented using the most basic array multiplier.

It's easy to see that all block products on the same diagonal path have the same significance in the final product. Therefore, we need to add these block products diagonally. The closer to the top

left corner, the more significance the block products possess, vice versa. This provides an intuitive way for us to turn off the right corner blocks since they have less significance of the final results.

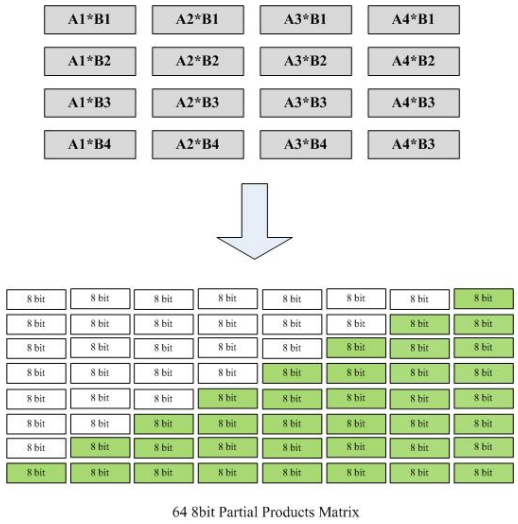


Figure 2 Partial product matrix generation for a 32 bit multiplier

2.2 Diagonal addition (Partial Product Accumulation)

As conventional multiplier does, there are two styles of diagonal addition, which are array style and tree style. According to A. Bellaouar etal, array structures are more suitable for low power applications since the excess wiring is likely to consume more power in tree structures[10]. In our case, we implement both the array and tree structure, the simulation results shows that our combined architecture can be faster and less power consuming than a conventional 32-bit Wallace tree structure. For the 8-bit multiplier, an array structure demonstrates both lower power and low latency. See Figure 9(a).

3. CIRCUIT IMPLEMENTATION

3.1 Circuit Description

We implement our design in Cadence 5.0 environment with NCSU Free PDK 45 nm technology. Figure 3 shows the top level structure of our design.

3.1.1 Block Multiplication

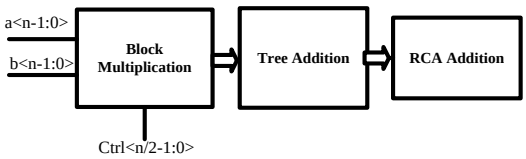


Figure 3 Top level structure

Each n/k block is further divided into n/(2k) blocks until each block contains 2 bit inputs. Then the 2 bit multiplier will calculate the small partial products and accumulate recursively until it is the partial product of the n/k block. As shown in Figure 2, the 32 bit multiplier divided into 4 blocks for each multiplicand will end up with 64 8-bit partial products. The lower left triangle colored in green indicates that these blocks could be potentially turned off to achieve power savings.

3.1.2 Wallace Tree Accumulation

Tree addition is the process of compressing the partial products. Instead of having 64 1-bit partial products for conventional 8-bit multiplier, we just have 16 4-bit partial products. This is shown in Figure 4.

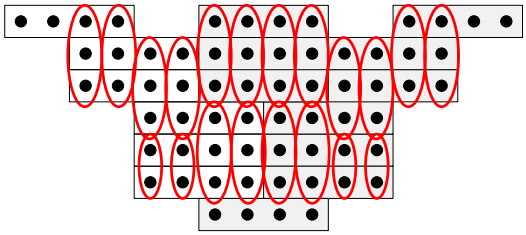


Figure 4 Tree Addition

We follow the same way as Wallace tree does to compress the partial products. 3 to 2 compressors and 2 to 1 compressors are used.

3.1.3 RCA Addition

After several steps of block partial products compressing, there are just two levels of partial products left. We use RCA Addition block to add the two levels and get the final products. The block is simply a ripple carry adder. This adder can be replaced as prefix adder. Considering the power consumption, we choose the ripple carry adder. By utilizing the inversion property of ripple carry adder, we can achieve lower power consumption. This is shown in Figure 5.

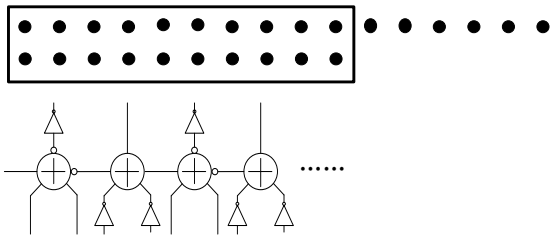


Figure 5 RCA Addition Stage

4. LOW POWER STRATEGY

4.1 Control Logic

As we mention in Section 3, each LSB block has control logic, we implement the idea of coarse power gating to enable or disable the corresponding blocks. The implementation is shown in Figure 6. The DEN signal controls the operational mode of the 2-bit multiplier. Table 1 shows the truth table of the control logic. When the multiplier block is in idle mode, the output is floating. To avoid this, we add NMOS transistor for each output to force the output to ground. In this way, we can still get the expected results without power penalty.

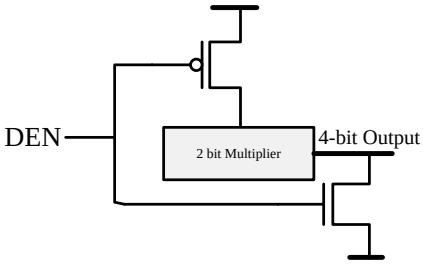


Figure 6 Implementation of coarse power gating

Table 1. Control logic Truth Table

DEN	Status of Multiplier Block
0	Operating
1	Idle

4.2 Sizing

Size is an important factor of delay and power consumption. To optimize the design, we pick the size of all the transistors properly. We have a large numbers of compressors and adders. Figure 7 shows the size of the PMOS in mirror adders and the energy-delay product plot. From the result, we pick all the PMOS in critical path 1.6 times the unit transistor to get the optimized energy delay product.

4.3 Other low power strategy exploration

In this work, we attempt to explore several other ways for low power design, like zero detection and voltage scaling. The idea of zero detection is detecting the all-zero block and turning off the corresponding blocks to save power. But our simulation results show that this will bring no benefit but overhead. The same as voltage scaling, the additional level shifters bring large overhead.

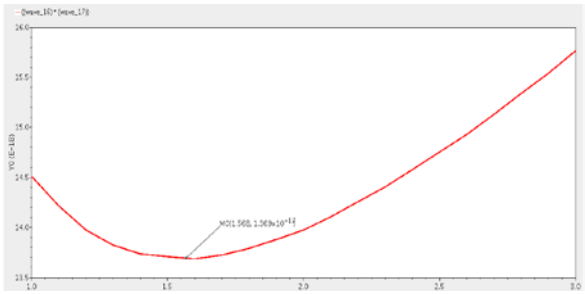


Figure 7 Size of PMOS vs. Et Product

5. RESULTS AND ANALYSIS

5.1 Power Simulation

The power-aware block imprecise tree multiplier (PABITM) is simulated under supply voltage 1.1 V and the Monte Carlo Simulation in Ocean Script is ran in Cadence Spectre simulator. Figure 8 shows the average power consumption of our 8-bit design compared with conventional 8-bit Wallace tree multiplier. The results show our multiplier without any truncation can achieve 1.5 % power saving, as the numbers of blocks turned off increase, the power decreases dramatically. Overall, the PABITM can achieve about 52% power saving with 10 blocks truncated.

5.2 Design Space Exploration

Shown in Figure 8 is the design space exploration plot for 8-bit multiplier design. m is the overall numbers of blocks, xb is the numbers of blocks turned off. The simulation results show 8-bit array multiplier can achieve both power saving and speedup, and our design consumes less power but slower than Wallace tree. While 32-bit multipliers design exploration plot in Figure 9 shows that our design has a great advantage over conventional Wallace tree multiplier both in power and delay. Overall, we can achieve about 50% power saving and 10% speedup.

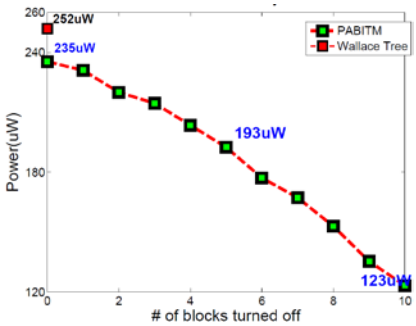


Figure 8 8-bit PABITM Power Fidelity Tradeoff

5.3 Error Characteristic Analysis

Error characterization for imprecise circuits is important because it determines what kind of applications that the multiplier can be used under different precision configurations. Figure 10 shows the error distribution of the multiplier under different precisions. The errors are represented by PMFs(Probability Mass Functions). They-axis represents the error frequency, and the x-axis represents the error magnitude[11]. The PMFs are centered at 0 representing 100% accurate result. The right half represents the positive error magnitude while the left half stands for the negative error magnitude. When the number of blocks tuned off is small, the imprecise multiplier falls into the category of a frequent small magnitude circuit according to Huang[12]. When the number of blocks turned off becomes larger, the error magnitude also becomes larger. Therefore, the multiplier becomes frequent large magnitude imprecise circuit. Depending on how error sensitive an application is, the multiplier can be configured for the corresponding application and provide the best energy and fidelity tradeoff.

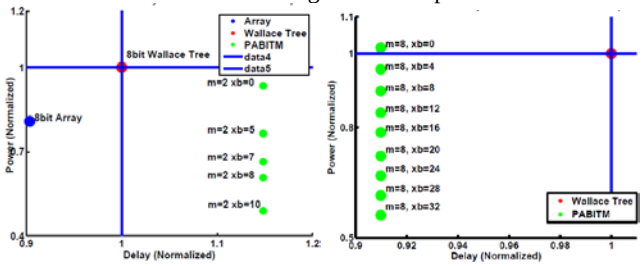
5.4 JPEG Compression Case Study

A case study on JPEG image compression was performed. We used the libjpeg-8d “lossy” compression library from Linux from scratch group [13]. There are a total of 11 multipliers used in the compression algorithm, and we replaced all 11 multipliers with our design. The Structural SIMilarity (SSIM) index is used as a quality measure of the compressed image compared to the orginal image[14]. The resulting images in Figure 11 show that the use of our imprecise multiplier only block the high frequency components and acting as a low pass filter therefore blurs the image with small penalty on image structure and quality.

6. CONCLUSIONS

This paper demonstrates a novel architecture of implementing a recursive block based power aware imprecise multiplier for

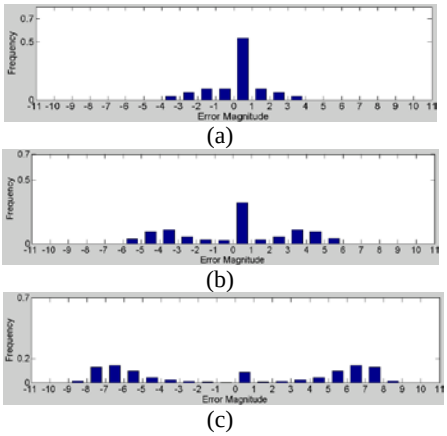
configurable low power and high performance. Fidelity is introduced as an additional design knob that provides the



(a) **(b)**

Figure 9 (a) 8-bit and (b) 32bit Multiplier design space exploration (m: number of bits for each block; xb: number of blocks turned off)

capability for more aggressive power savings. Different low power design techniques in both architecture level and circuit level were discussed. And the result shows that the 32 bit implementation can achieve up to 47% power savings with 10% improvement in performance. And the JPEG image compression case study demonstrates the promising application domains that this design can be well applied. Some future work may include implementing the dynamic re-configurability of the circuit so that the multiplier could adjust power and fidelity tradeoff during the run time. More aggressive low power techniques could be explored such as data preprocessing which includes zero detection and booth encoding. This architecture is also ready to be extended to floating point multipliers.



(a)

(b)

(c)

Figure 10 Error Frequency vs. Error magnitude distribution (8 bit, block size = 4); (a) # of turn off block = 1; (b) # of turn off block = 2 ;(c) # of turn off block = 5;

Figure 11 JPEG Image Compression Case Study

7. ACKNOWLEDGMENTS

We would like to sincerely thank Prof. Mircea Stan and Prof. John Lach for their advices.

8. REFERENCES

[1] Xiaoxiao Zhang, Amine Bermak, Farid Boussaid, *Power Optimization in Multipliers Using Multi-Precision Combined with Voltage Scaling Techniques*, in 1st Int'l Symposium on Quality Electronic Design-Asia. 2009

[2] Jieh-Hwang Y.; L-R Dung and C.Yuan, Shen, *Design of Power-Aware Multiplier with Graceful Quality-Power Trade-Offs*, IEEE International Symposium on Circuits and Systems (ISCAS 2005), Vol. 2, pp:1642 - 1645, May 2005

[3] M. S. K. Lau, K. V. Ling, and Y. C. Chu, *Energy-aware probabilistic multiplier: Design and analysis*, in Proceedings of the 2009 International Conference on Compilers, Architecture and Synthesis for Embedded Systems, 2009, pp. 281-290.

[4] Jiun-Ping Wang, Shiann-Rong Kuang and Yuan-Chih Chuang, *Design of Reconfigurable Low-Power Pipelined Array Multiplier*, Communications, Circuits and Systems Proceedings, Volume 4, Page(s):2277 - 2281, 25-28. June 2006.

[5] Xiaoxiao Zhang, Amine Bermak, Farid Boussaid, *Dynamic Voltage and Frequency Scaling for Low-power Multi-precision Reconfigurable Multiplie*, Proc. of 2010 IEEE International Symposium on Circuits and Systems, pp. 45-48, Paris, 2010.

[6] C.-H. Chang, R. K. Satzoda, and S. Sekar, *A novel multiplexer based truncated array mutliplier*, IEEE International Symposium on Circuits and Systems, vol. 1, pp. 85-88, 2006.

[7] Wei Ling; Savaria, Y., Variable-precision multiplier for equalizer with adaptive modulation, Circuits and Systems, 2004, MWSCAS '04, Volume 1, Page(s):I-553 - 556, July 2004

[8] Z. Huang and M. D. Ercegovac , *High-performance low-power left-to-right array multiplier design*, IEEE Trans. Comput., vol. 54, no. 3, pp.272 -283 2005

[9] T.K. Callaway, and E.E. Swartzlander Jr., *Low Power Arithmetic Components. in Low Power Design Methodologies*, Rabey, J. and Pedram, M., eds., pp. 161-198, Norwell, Mass: Kluwer Academic Publishers, 1996

[10] A. Bellaouar, and M.I. Elmasry, *Low-Power Digital VLSI Design, Circuits and Systems*, pp. 442-450, Norwell, Mass: Klu- wer Academic Publishers, 1995

[11] J. Huang, J. Lach and G. Robins, "*Analytic Error Modeling for Imprecise Arithmetic Circuits*", Silicon Errors in Logic - System Effects (SELSE), 2011

[12] J. Huang and J. Lach, "*Exploring the fidelity-efficiency design space using imprecise arithmetic*," 16th Asia and South Pacific Design Automation Conference (ASP-DAC 2011), pp. 579–584, Jan. 2011.

[13]http://www.linuxfromscratch.org/blfs/view/svn/general/libjpeg.html

[14] https://ece.uwaterloo.ca/~z70wang/research/ssim/